The Full ALU

Author:

Patrick Peng

Design:

Adder

To build a non-RCA adder, I use a 32-bit CSA consisting of three 16-bit RCA.

For each 16-bit RCA, I use 16 full adder.

To achieve the addition and subtraction in one adder, I use a 2 to 1 mux to choose from B and B' (in verilog,

Bitwise Operation

To achieve 32-bit bitwiase and/or, I use 32 and/or gates. Each gate takes one bit from the input (A , B)

Shift Operation

For barrel shifter, I use the same logic as the slides given by Prof. I implement it with 1-bit 2:1 mux. For

isNotEqual

I use xnor gate and and gate to see if A and B are the same. Firstly, I use xnor gate (A, B as inputs) to get

isLessThan

The result is realated to the subtract operation. Moreover, there are special case related to the overflow. S

overflow

```
The overflow is generated through four conditions: positive + positive, negative + negative, positive - negat
```

Behavior

```
If ctrl_ALUopcode = 00000, data_result will be the result of addition;

If ctrl_ALUopcode = 00001, data_result will be the result of subtraction.

If ctrl_ALUopcode = 00010, data_result will be the result of bitwise_and.

If ctrl_ALUopcode = 00011, data_result will be the result of bitwise_or.

If ctrl_ALUopcode = 00100, data_result will be the result of SLL.

If ctrl_ALUopcode = 00101, data_result will be the result of SRA.

It has four output: 32-bit result (data_result),1-bit overflow (overflow), 1-bit isNotEqual (isNotEqual) and
```